Automated Learning of Loop-Free Alternate Paths for Fast Re-Routing

Wouter Tavernier*, Dimitri Papadimitriou†, Didier Colle*, Mario Pickavet*, Piet Demeester*
*Department of Information Technology (INTEC), Ghent University – IBBT
Gaston Crommenlaan 8, 9050 Gent, Belgium
E-mail: {wouter.tavernier, didier.colle, mario.pickavet, piet.demeester}@intec.ugent.be
†Alcatel-Lucent Bell, Antwerp, Belgium
E-mail: dimitri.papadimitriou@alcatel-lucent.be

***Abstract*— Upon failure detection, IP networks need to reconfigure their routing and forwarding tables. Typically this task is executed by routing protocols such as Open Shortest Path First (OSPF). However during these short re-convergence periods, transient loops can occur, resulting in datagram loss of affected traffic flows. Several techniques have been developed to reduce the resulting datagram loss or to avoid this situation. However these approaches are not always able to cover all potential routing loops and can involve additional configuration. In this paper we suggest an alternative approach which relies on configuring loop-free alternate forwarding entries using learning techniques to reduce the configuration and setup effort, while still offering high speed switchovers upon failure events with minimal datagram loss resulting of transient loops. We show in a simulation environment that improved results can be obtained with respect to the number of link failures that can be covered, the resulting probability on having cycles in the alternate routing, the resulting quality of the alternate routing, and the induced communication cost of the learning procedure.**

I. INTRODUCTION

Connection-less IP networks independently decide how to forward received packets or datagrams. The information determining how they forward these packets (i.e. which outgoing interface and next hop they will take) is stored in their local Forwarding Information Base (FIB). The FIBs comprise (forwarding) entries that are derived from the information exchanged by link-state routing protocols such as Open Shortest Path First (OSPF, [1]). These protocols discover the local topology (link states) and distribute the discovered information over the network using Link State Update messages. As a consequence, every router can independently compute the shortest routing paths towards other nodes in the network. Exchanges of link state routing information resulting from topology changes (dynamic reaction to topological changes due to, e.g., link/node failures) lead to the re-computation of the routing paths and reconfiguration of the corresponding FIB entries (re-convergence), as well as the update of the corresponding routing and forwarding entries (note that these steps outline the IGP[1] re-convergence process). However, as every router performs the routing path computation independently of other routers, transient (micro-)loops may be formed during the periods when a network is re-converging due to inconsistent FIB entries. This problem is inherent to any asynchronous distributed routing protocol and caused by inconsistent FIB entries resulting from the propagation time of the routing updates as well as the time needed to re-compute and distribute FIB entries.

Packets which are trapped into transient loops, never reach their destination and are simply lost after TTL expiration. Prior work [2] has demonstrated that these loops can take hundreds of milliseconds. Therefore, our goal is to minimize the re-routing time: the time needed for each node, after the occurrence of a topological change, to use updated FIB entries -*without relying on the full IGP re-convergence*- along loop-free alternate paths for the maximum number of destinations. The three-fold objectives of the paper (and of fast-rerouting techniques in general) are:

- Maximize the percentage of links (or nodes) that can be fully protected (i.e., for all destinations)
- Maximize the percentage of destinations that can be protected for all link (or node)
- Minimize the stretch increase on the routing paths between source and destination.

The proposed fast-rerouting technique relies on the avoidance of transient loops by detecting them before failure occurrence. More precisely, it operates following three main steps. Initially, each node determines its loop domain with respect to other (destination) nodes. The loop domain is determined by the set of nodes for which the loop-free neighbor criteria is not verified along certain alternate routing paths before occurrence of topological change (when traffic forwarded by node u and directed to destination t arrives at node v that forwards this traffic along a path that reaches node u, i.e., v is a not loop-free neighbor of u). Then, the detecting node selects an alternate routing path that ensures loop-freeness up to loop domain boundaries by instantiating an alternate forwarding entry on each intermediate node (pointing to the loop-free neighbor). Upon failure occurrence, the node triggers that loop-free alternate path (when traffic from u directed to t arrives at v, v does not forward traffic along a path that reaches node u, i.e., v becomes a downstream neighbor of u).

[1] The term Interior Gateway Protocol (IGP) refers to any link state routing protocol running within a routing system.

This paper is structured as follows. In Section II, related work is described with respect to fast re-routing techniques. Next, in Section III we outline the main contribution brought by the proposed approach. Section IV provides a detailed description of the proposed technique including a learning approach finding nodes out of the loop domain of each node and the computation of the loop-free alternate path. Section V details the experimental results we have obtained by simulation when running these procedures on topologies representative of core networks to which the proposed technique would typically apply. Finally, Section VI formulates the conclusions of the paper and some suggestions for future work.

II. RELATED WORK

Fast re-routing (or fast repair) techniques can be classified into the following three basic categories (see [3]).

A. Equal cost multi-paths (ECMP)

ECMP [4] can be used when a set of two or more paths towards the same destination d is available. Assuming one of them doesn't traverse the failure, that alternate path can be used as repair path.

B. Loop-free alternate (LFA) paths

A Loop-Free Alternate path [5] exists when a direct neighbor of the router adjacent to the failure has a path to the destination that can be guaranteed not to traverse the failure (loop-free neighbor condition). The average coverage on common networks (that is strongly dependent on the topology) shows variations from 60 to 90%. Indeed, when a link or a node fails, only the neighbors of the failure are initially aware that the failure has occurred and only neighboring node to the failure repair the failure. These repairing routers have to steer datagrams to their destinations despite the fact that most other routers in the network are unaware of the nature and the location of the failure. A common limitation in most of the base LFA mechanism is an inability to indicate the identity of the failure and to explicitly steer the repaired datagram round the failure. Consequently, the extent to which this limitation affects the repair coverage is topology dependent. An advanced LFA solution [6] consists in sequencing the FIB updates either spatially (topologically ordered FIB update from far-end to the near-end neighbor contiguous to the failure) or temporally (timely synchronized FIB updates). For instance, ordered FIB update provides 100% loop-free convergence at the expense of a FIB update time proportional to R x MAX_FIB, where, R is the max (hop) length among paths to edge r used to reach destination t (downstream SPF neighbor prior to the failure) and MAX_FIB is a network-wide constant that reflects the maximum time T_{max} required to update a FIB irrespective of the change required. Hence, degrades proportionally to the path length i.e. FIB updates are actually committed at the near-end after reception of a completion message traveling back from the source of max (hop) length among path to edge r used to reach destination t. This solution is not considered outside network maintenance operation as it suffers from slow activation

C. Multi-hop repair paths

When there is no feasible loop-free alternate path it may still be possible to locate a router, which is more than one hop away from the router adjacent to the failure, from which traffic will be forwarded to the destination without traversing the failure. Multi-hop repair paths are more complex both in the computations required to determine their existence, and in the mechanisms required to invoke them. Multi-hop repair paths techniques can be further classified as:

i. Mechanisms where one or more alternate FIBs are pre-computed in all routers, and the repaired datagram is instructed to be forwarded using a "repair FIB" by some method of per-datagram signaling involving, e.g., the detection of a "U-turn" [7].

ii. Mechanisms functionally equivalent to a loose source route that is invoked using the normal FIB. These include tunneling-based approaches [8] that consist in "by-passing" the topology change by pre-configuring tunnel whose path is not affected that change. There are multiple variants of "tunnel-based solutions": single-sided (near-end or far-end), double-sided (near-end and far-end), and distributed (tunnel segments). They all suffer from the same problems: i) computational complexity, ii) tunnel pre-configuration and maintenance, and iii) impact on forwarding plane. Thus, they all involve a high degree of configuration for tunnels that in turn decrease the forwarder performance. Other mechanisms such as the Not-Via technique [5] employ special addresses that are installed in the FIBs together with pre-computed routes that avoid certain components of the network. This technique encapsulates the datagram to an address that explicitly identifies the network component that the repair path must avoid. This produces a mechanism that always achieves a repair, provided the network is not partitioned by the failure.

III. OUR CONTRIBUTION

As outlined in the previous section, several fast path repair/fast re-routing techniques already exist. Some of them are used in operational networks such as base Loop-Free Alternates (LFA) and Equal Cost Multi-Path (ECMP). They all aim to address the objectives detailed in the introductory section of this document. Our contribution is threefold: i) the proposed technique relies on distributed learning of the loop-domain at each node and "best-alternate path" to a given destination. Both can either be performed on-line or by mining the link-state routing topology and the routing table (RT) entries; ii) the proposed re-routing scheme does not assume modification of the link-state routing protocol operations outside of the transient re-routing periods (as

alternate forwarding entries take local precedence over default IGP routing entries). Once, the IGP has re-converged unflagging datagrams leads to the use of the primary path entries; iii) the coverage of the proposed re-routing scheme is almost 100 %.

IV. Automated Learning of Loop-Free Alternates

A. Assumptions

The proposed approach aims to accelerate the re-routing of traffic along loop-free alternate routing paths in link state routing networks. Upon failure occurrence, the failure detection technique is assumed to provide local information. Failure information propagation does not rely on associated fast failure notification protocol (operating next to the link-state IGP) or IGP parameter tuning. The only condition for our approach to be operational is that the loop domain's diameter is smaller than the flooding domain of the IGP. Otherwise, the technique resumes as a best exit node selection to avoid loops inside the IGP routing domain but then relies on neighboring domains for the alternate path to remerge with the primary path (outside the loop domain).

B. Preliminaries

The network topology is modeled by a weighted undirected graph G = (V, E, ω) with positive edge cost ω, where V is the set of vertices or nodes (|V| = m) and E is the set of edges or links (|E| = n). A non-negative cost function ω: E $\rightarrow Z^+$ associates a cost $\omega_{u,v}$ to each link (u,v) $\in$ E. For s, t $\in$ V, let d(s,t) denote the cost of the path p(s,t) from s to t in G, where the cost of a path is defined as the sum of the costs along its edges. We first introduce the following distinction:

- For the pair s, t $\in$ V, s $\neq$ t, if there exists a vertex u adjacent to vertex s, (i.e., edge (s,u) $\in$ E(G)) such that d(u,t) < d(s,u) + d(s,t), i.e., u is a loop-free neighbor of s to destination t, then the path (v_0(=s), v_1, ..., v_m(=t)) is a loop free alternate path where $\forall i : d(v_i,v_m) < d(v_{i-1},v_i) + d(v_{i-1},v_m)$.
- For the pair s, t $\in$ V, s $\neq$ t, if there exists a vertex u adjacent to vertex s, (i.e., edge (s,u) $\in$ E(G)) such that d(u,t) < d(s,t), i.e., u is a downstream neighbor of s to t, then the path (v_0(=s), v_1, ..., v_m(=t)) is a distance decreasing downstream path where $\forall i : d(v_i,v_m) < d(v_{i-1},v_m)$. As a particular case, neighbor u of node s is the downstream SPF neighbor of s for destination t, if node u provides the shortest path to t according to a shortest-path first (SPF) routing scheme.

Note that the set of distance decreasing downstream paths is a subset of the set of loop-free alternate paths meeting the condition $\forall i : d(v_i,v_m) < d(v_{i-1},v_m)$.

We define the loop domain of node u $\in$ V(G) as the set of node B(u) such that if a path p(s,…,u,…,w,…t) traverses node u and then node w it will loop back via node u before reaching destination t, i.e., w does not sit along a loop-free alternate path to destination t from node u.

C. Steps and Mechanisms

The proposed fast re-routing approach (ALFA) comprises three main steps:

Step 1: each node u determines its loop domain B(u) with respect to each destination t that it can reach (as indicated by its routing table entries). For this purpose, node u sends a probe message towards destination t on the interface directed to one of its non-shortest path from u to destination d. If the message returns to u (source of the probe message) the message didn't reach a node v located outside of the loop domain. We refer to such node v as a loop-free node (LFN).

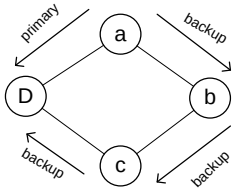
Step 2: determine a node v located outside the loop domain of node u for destination t and that sits along a non-shortest path towards destination t. Node v is referred to as the loop-free node (LFN) and the path (u…,v,…,t) as the loop-free alternate path (or more synthetically p(u,v,t)). Inside the loop-domain B(u) of node u, along the non-shortest path that is selected as the loop-free alternate path and on which the probe message sourced at node u is forwarded, alternate forwarding entries are configured for that destination t. Indeed, the default forwarding entries at these nodes for destination t refer to a path that traverses node u. More precisely, for $\forall w \in B(u)$ | node w does not verify the loop-free condition, the path p(w,t) includes node u, i.e., p(w,u,t). When the probe message reaches node v, that message is returned to node u with the indication that no FIB entry configuration is required to reach destination t (node v verifies the loop-free condition: d(v,t) < d(u,v) + d(u,t). Note that with the BFS+ technique (as documented in Section IV.C), the loop-free alternate path p(u,v,t) is the non-shortest path that differs the most from the shortest path (considered as the primary path) before failure of a link incident to u along the primary path from u to t, p(u,t) | v $\notin$ p(u,t).

Step 3: activation upon failure detection: upon failure detection by node u (assume, e.g., the failure of one of the links incident to node u along its primary path towards destination t), the loop-free alternate path is activated. The action of activation by node u of its loop-free alternate path p(u,v,t) refers to the triggering operation of the alternate forwarding entry along the loop-free alternate path inside the loop domain of node u, B(u). The alternate forwarding entries are triggered from the reception of datagrams including as indication in their header that these datagrams were re-routed by node u along the loop-free alternate path. Activation of the alternate forwarding entries is performed until reaching node v. Outside of the loop-domain of node u, datagrams remain flagged but without triggering any action at the nodes traversed by these datagrams (the alternate and the primary forwarding entries are indeed identical). This condition is sufficient to guarantee that the path p(v,t) followed by the datagrams leaving the loop-domain is loop-free as long as the path p(v,t) is the distance decreasing SPF downstream path to destination t (the path p(v,t) does not re-enter the loop domain of node u). When exiting the local

routing domain (i.e., the link state routing protocol flooding domain), the datagrams flagged by the re-routing node u are unflagged by the boundary node of the domain.
The next paragraphs of this section explain each of these steps together with a description of the corresponding procedures.

D. Forwarding model of routers

A router consists of a Routing Information Base (RIB) and a Forwarding Information Base (FIB). In the context of this paper the terms RIB and routing table are used equivalently since we assume that a single routing protocol is running in each routing domain. The FIB stores forwarding entries each comprising the outgoing interface to be taken by individual datagrams for a given destination prefix. The router model we use in this paper, allows to store as part of the FIB, an alternate forwarding entry for any given destination prefix. The use of the alternate entry is triggered by the indication of a flag (bit) in the header of an incoming datagram (to be decided in which field), further referred to as the *alternate flag*. Datagrams are marked with this flag, from the moment a failure is noticed on the link towards the next hop according to the primary forwarding entry.
In our router model, the forwarding decision is also conditioned on the incoming interface, which implies that the alternate entry for a given destination prefix can be different for datagrams arriving at interface x, compared to those arriving at interface y in a given router. This interface-dependence allows us to keep using shortest path routing on the primary forwarding entries. To ensure that the alternate forwarding entries have node-wide significance, the identifier of the triggering node (that is the node that flags the datagram) should be known and stored at configuration time as part of the alternate entries and be included as part of the flagged datagram. This is illustrated in the figure below. The shortest path towards node D from node a and c is via their direct link. However, using node-wide significant alternate routing entries to node D enforces them to choose whether node a or node c is on the primary path.



If, the primary next hop of node u along its primary path to a given destination becomes unreachable due to a link or node failure, then i) the datagrams for that destination are flagged (as indicated before) and ii) the alternate forwarding entry for the interface corresponding to the failing link or node is chosen to forward the flagged datagrams along the alternate path. At node u, the use of the alternate forwarding entry must not result into flagged datagrams being sent back to node u (rule.1). Along the alternate path, flagged datagrams arriving from primary interface (i.e., the interface corresponding to the next hop as indicated in the primary forwarding entry) or more generally any interface if the identifier of the triggering node can be retrieved from the incoming datagram, the alternate flag will automatically trigger the use of the alternate forwarding entry to avoid looping behavior (rule.2). To avoid that the flagged datagrams loop back to node u, the proposed technique comprises a cycle-free alternate path computation technique. This technique is described in the next section.

E. Cycle-free alternate path computation

a) Initial FIB configuration

We initiate the Primary FIB (PFIB) of all nodes using the usual shortest-path computation techniques for (connection-less) link-state routing protocols such as OSPF or IS-IS. The alternate FIB (AFIB) stored at each node is initially a copy of the PFIB, using the same next hop for on all interfaces as the one determined by the shortest path calculation for the PFIB. This has one noticeable exception: the AFIB-entry corresponding to the primary forwarding entry is populated with the next hop according to the shortest path excluding the link indicated by the primary forwarding entry. We will refer to this entry as the Alternate Shortest Path entry (ASP entry). Note also that after configuration, the forwarding entries for which the primary and the alternate next-hop for the same destination are identical can be removed from the AFIB. Furthermore, FIB compression techniques (one entry for multiple prefixes) can be used to reduce the memory space used by the AFIB.

b) Alternate FIB configuration

As previously explained, once the moment a single failure is locally detected by a given router, its incoming datagrams toward the affected destinations are flagged, and the datagrams are forwarded according to the alternate forwarding entry (the ASP entry as defined here above). However, because downstream routers still forward flagged datagrams according to their locally computed shortest path, it is likely that the flagged datagrams will be looped back to the flag-originating-node (FON), causing a forwarding loop.
To avoid forwarding loop situations, we combine two techniques: i) the discovery of a node referred to as the loop-free node (LFN) which sits outside of the loop-domain of a given node with respect to a given destination, and the LFN is out of the loop-domain of the given node with respect to the LFN itself, and ii) the configuration of the AFIB-entries along the path towards the given LFN, this path is the one referred to as the alternate path. The loop-domain of a given node u for a given destination d is defined as the set of downstream nodes (with respect to the directionality of the traffic flow towards destination d) that forward incoming datagrams received from node u along a path that traverses node u. Once flagged, the datagrams reach the LFN, the path followed according to the rest of the AFIBs lead to the destination without looping back to the original node again.

F. Loop-domain detection using BFS+

As indicated earlier, in order to ensure a loop-free alternate path from a node s towards a destination d, the former needs to find a node (LFN) out of its loop-domain with respect to

d. For this purpose we devise an extended Breadth First Search method (referred to as BFS+). The recursive mechanism works as follows:

- Send a probe message towards d via all the neighbors of the node s (hop count diameter 1 from s).
- If all probe messages pass via the node s, mark the visited nodes, and repeat the procedure with the neighbors of the marked nodes (excluding the already visited nodes) until at least one probe message reaches destination d without passing via source node s. Upon the arrival of the probe message in d, the receiving node sends an acknowledgement message towards node s.
- When multiple LFNs are found within a given diameter from node s, the LFN is chosen such that the alternate routing path towards d has the lowest similarity with the primary path from s to d[2].

The BFS+ algorithm is illustrated on Figure 1. The loop-domain of node s is indicated with the circle with dotted lines, containing the nodes probe1 and probe2. Probe1 and probe2 are upstream nodes to s with respect to destination d. This implies that the shortest path of these nodes will always pass via node s. Because these nodes are within hop count diameter 1, BFS+ will first send probe via these nodes towards destination d. When the node s intercepts these probe messages, BFS+ triggers probe messages to be send from hub nodes on hop count diameter 2. Afterwards, the nodes probe3 and probe4 are tested. Both nodes forward the probe message to d without passing node s, and thus are LFNs. However, because the path taken from node probe3 differs more from the primary path compared to the path taken from node probe4 (which uses the same last link towards d), probe3 is elected as the LFN by the procedure.

G. Configuration of path to LFN

Once an LFN node is elected using the previously described BFS+ technique, the loop-free alternate path towards the LFN must be configured. This operation is realized by installing the alternate forwarding entries along the alternate routing path from node s to the LFN. For this purpose, the node s sets its forwarding entry towards the LFN as its alternate entry towards destination d. The same procedure is used as the indicated next hop(s) until the LFN is reached.

H. Alternate path usage upon failure detection

The ALFA-learning procedure executes the above LFN-detection and LFN-path-configuration process from all nodes towards all other nodes (destination).

When a node detects that the outgoing interface corresponding to the primary routing entry for a given destination is not available, based on a loss-of-signal event or a Hello-timer timeout (as in OSPF), the alternate forwarding entry towards the destination is used. This procedure will bring the packet to the LFN (as it was previously configured to do so), and from then on, shortest path routing entries will bring the packet from the LFN to the destination.

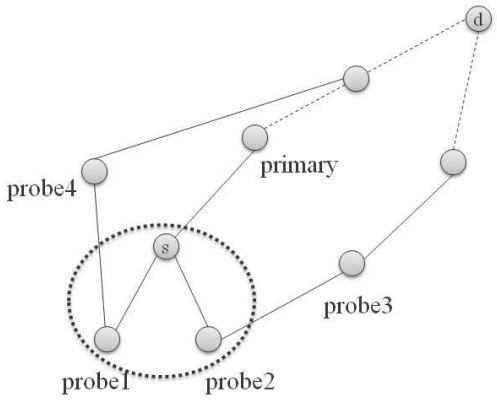


Figure 1 - Loop-domain detection example using BFS+

V. EXPERIMENTATION

A. Environment

A custom simulation environment was developed by means of Python/C++ libraries to benchmark the discussed cycle avoidance techniques on a number of different networks. The default routing behavior (primary routing entries) of the benchmarked networks follows shortest path routes as configured by a distributed link-state routing protocol such as OSPF. To obtain representative results, the computed routes were randomized and made independent between the nodes in the network. This implies that, if multiple shortest paths are available between two network nodes, every run will randomly choose a route, and configure the routing tables accordingly. This performed independently on every network node. Every experiment has been re-run 100 times with these randomized settings, and the reported quantitative results are averages over these runs.

Table 1 - Network topologies

Network	***Nodes***	***Links***	***Degree***		
			Min	***Avg***	***Max***
Abilene	11	14	2	2.55	3
Nobel-us	14	21	2	3.00	4
Nobel-ge	17	26	2	3.06	6
Garr	22	36	2	3.27	9
Nobel-eu	28	41	2	2.93	5
Geant2	30	47	2	3.13	8
Renater	36	49	2	2.72	7
Cost266	37	57	2	3.08	5
Germany50	50	88	2	3.52	5
Xwin	57	77	2	2.70	6

[2] The similarity of paths from two nodes towards a third, can be measured by actively storing temporary forwarding states during the probing process, or using traceroute measurements as is performed in [10]

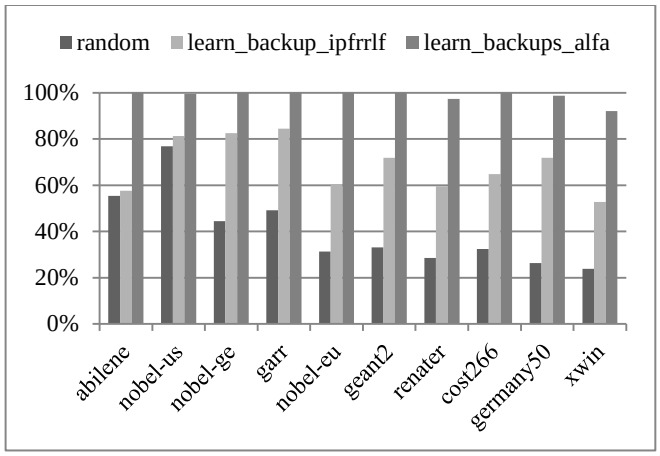


Figure 1 – Percentage of link failures covered

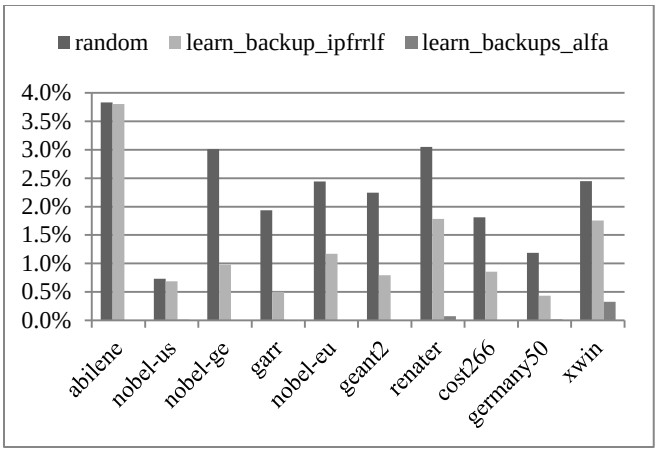


Figure 2 – Routing cycle probability upon link failure

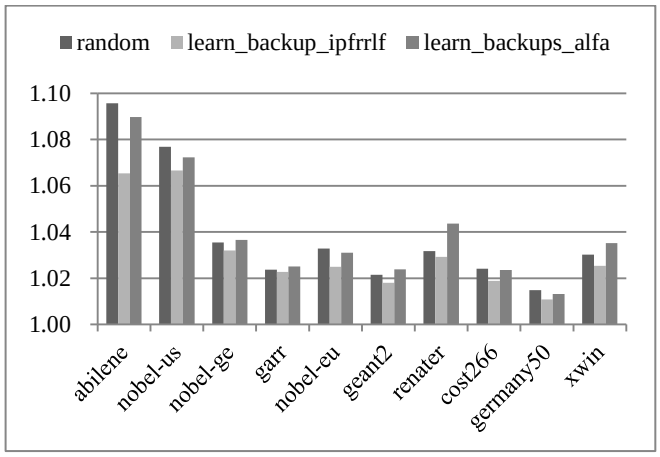


Figure 3 - Stretch comparison

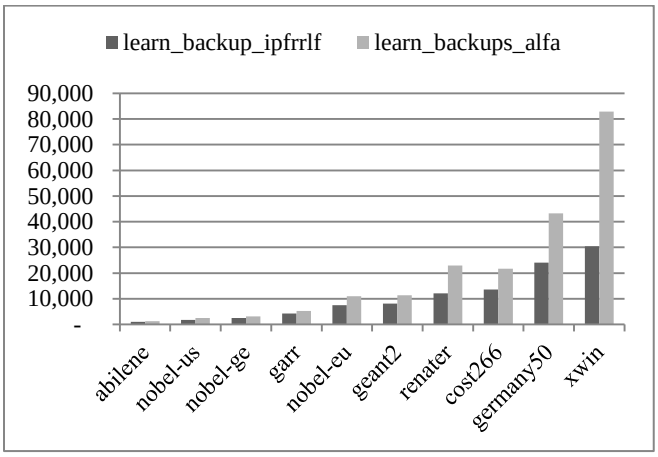


Figure 4 - Number of probing messages needed to converge

B. Network topologies and network traffic

A set of 10 representative reference networks was used for evaluating the described techniques. Most of these networks are known for research purposes (e.g. [11]), or are research networks themselves. The number of nodes of these networks ranges between 11 and 57 nodes, and their node degree is in the range [3,9]. For some of these topologies, single connected nodes have been removed, because alternate routing paths are not possible for these anyhow. The properties of the reference networks are summarized in Table I.

C. Benchmarked techniques

The techniques with the following labels were benchmarked:

1) random

This technique refers to the configuration of usual shortest path routing entries as performed by a link-state protocol such as OSPF, augmented with random alternate entry routing entries (the only requirement is that the alternate next outgoing interface is different from the primary outgoing interface).

2) learn_backup_ipfrrlf

The scheme technique refers to the configuration of Loop-Free Alternates (also referred to as FRR-LFA) as backup entries as discussed in Section II.B [5], if they are available. Finding a loop-free alternate entry is performed by probing the paths from nodes' neighbors to check if they loop-back towards the originating nodes.

3) learn_backups_alfa

Here, alternate forwarding entries are configured using the proposed ALFA technique from Section IV, which finds Loop-Free alternates using the BFS+ method. In this case, the LFN is chosen within the diameter of the closest node out of the loop-domain, having a path towards the targeted destination which differs maximally within the probed neighborhood.

D. Performance measurements

This section discusses the performance of the mentioned techniques with respect to their ability to cover link failures (coverage), their consequences on the resulting length of the backup paths (stretch), their communication cost for learning adequate entries, and their sensitivity with respect to network characteristics.

1) Coverage

For each topology (see Table I), every possible single link failure was simulated. When a configured alternate forwarding entry (using one of the three presented techniques) is not able to recover the connectivity between all network nodes for a given link failure, the link is considered to be uncovered. The percentage of links which cannot be fully recovered upon link failure is denoted as the link failure coverage of the technique (the complement of this percentage denotes the percentage of links which can induce cycles among at least one source destination pair). Figure 2 depicts the link failure coverage of all considered re-routing schemes on all evaluated networks. From this figure, we may observe that provisioning randomly alternate entries is (as expected) is only able to cover 20 to 50 percent of the link failures. FRR-LFA is able to cover larger percentages of link failures, typically between 50 and 80 percent. The ALFA-technique which we propose is able to cover almost all link failures, or at least 95 percent.

The probability that a link failure will cause a cycle when selecting the alternate routing path between a random pair of nodes can be calculated, by evaluating the connectivity between all pair of nodes, for all possible single link failures. Figure 3 shows the resulting end-to-end cycle probability induced by a single link failure for the experimented networks for all the considered schemes. One can observe from this figure that the probability that a cycle occurs between a given pair of nodes is highest when only providing random alternate entries, and lowest – close to zero – when using the ALFA scheme. For small networks, the difference between provisioning random alternate forwarding entries and FRR-LFA is negligible.

2) Stretch

The previous metric measured the quality of the protection techniques in terms of the proportion of link failures they can potentially (fully) recover from. However, this metric doesn't give us information on the quality of the resulting alternate routing paths with respect to the path length. It may be expected, that higher recoverability could have a detrimental influence on the resulting path length. To assess this assumption, we calculate and depict the average stretch of the alternate routing paths of all techniques in all networks in Figure 4. The stretch metric indicates the ratio of the length of the alternate routing path (in hop count) vs. the length of the shortest routing path when no failure occurs. When the alternate routing path has the shortest length, the stretch is equal to 1. The average stretch calculates the average ratio over all resulting path lengths. Figure 4 illustrates that the length of the selected alternate paths taken out of all experimented techniques is at worst 10 percent longer than the (primary) shortest path between two nodes. FRR-LFA in general uses the shortest backup paths. This may be a consequence of the fact that only the first hop upon the failure is different from the primary shortest path in the network, while the ALFA technique may use longer detour paths to ensure that the packet is out of the loop-domain of the failure detecting node. This explains the higher stretch values for the ALFA technique.

3) Communication cost

Finding adequate alternate routing paths ensuring that no cycles occur requires some probing and learning activity in the network. Clearly techniques relying on probing lead to a cost with respect the number of probing messages. Both IPFRR–LFA and the ALFA technique involve probing: the first probes for a loop-free alternate neighbor, the second probes for a node out of the loop-domain of the failure detecting node. Figure 5 depicts the number of probing messages that were needed before the required techniques converged. The results illustrate that ALFA has probing communication cost between 20 percent (for the smallest networks) and 270 percent (for the largest) higher networks.

VI. CONCLUSION

In this paper, we proposed an alternative (learning) method for populating alternate routing entries. The resulting technique is able to avoid almost 100 percent of potential routing cycles upon the occurrence of single link failures in a given set of representative reference networks. We showed that this had low impact on the quality of the resulting backup paths, which were at most 10 percent longer than the shortest paths in the fully operating network. Future work could focus on reducing the induced communication cost of learning adequate alternate entries. Using the spatial correlation between several nodes, clustering techniques could correlate groups of destinations allowing common LFNs along their alternate path.

ACKNOWLEDGMENT

This work is supported by the European Commission (EC) Seventh Framework Programme (FP7) ECODE project (Grant n°223936).

REFERENCES

[1] Moy, J., OSPF Version 2, RFC 2328, Internet Engineering Task Force, 1998

[2] Francois, P., Achieving Sub-Second IGP Convergence in Large IP Networks, ACM SIGCOMM Computer Communication Review, July 2005.

[3] Shand, M., IP Fast Reroute Framework. RFC 5714, Internet Engineering Task Force, 2010.

[4] Hopps, C., Analysis of an Equal-Cost Multi-Path Algorithm, RFC 2992, Internet Engineering Task Force, 2000

[5] Atlas, A., Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286, Internet Engineering Task Force, 2008.

[6] Francois, P., Loop-free convergence using oFIB, Internet Draft, Internet Engineering Task Force, 2011

[7] Atlas, A., U-turn Alternates for IP/LDP Fast-Reroute, Internet-Draft, Internet Engineering Task Force, 2006.

[8] Bryant, S., IP Fast Reroute using tunnels, Internet Draft, Internet Engineering Task Force, 2007.

[9] Shand, M., IP Fast Reroute Using Not-via Addresses, Internet Draft, Internet Engineering Task Force, 2011.

[10] Hu, N., Quantifying Internet end-to-end route similarity, Passive and Active Measurement Conference, 2006

[11] Orlowski, S., SNDlib 1.0—Survivable Network Design Library, Networks, Special Issue: Network Optimization (INOC 2007), Volume 55, Issue 3, pages 276–286, Wiley, May 2010